08

Haga lo que hicimos en aula: mapeando relacionamientos

Ha llegado el momento de que sigas todos los pasos que he dado durante esta lección. En caso de que ya lo hayas hecho, excelente. Si aún no lo ha hecho, es importante que realice lo que se vio en los videos para que pueda continuar con la siguiente lección.

- Lo primero que vamos a realizar es organizar el código delegando la responsabilidad de instanciar el EntityManager a una clase utilitaria cuya única función será instanciarlo

```
public class JPAUtils {

    private static EntityManagerFactory FACTORY = Persistence.

    public static EntityManager getEntityManager() {
        return FACTORY.createEntityManager();
    }
}
```

COPIA EL CÓDIGO

- Luego vamos a realizar el mapeamiento entre entidades esta relación puede ser entre una entidad y un Enum o entre dos relaciones.
- Para relacionamientos entre entidades y Enum:

Se debe modificar la entidad y utilizar la anotación adecuada para el tipo Enum de esta forma JPA sabrá reconocer el elemento Enum

```
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
private Long id;
private String nombre;
private String descripcion;
private BigDecimal precio;
private LocalDate fechaDeRegistro = LocalDate.now();
@Enumerated(EnumType.STRING)
private Categoria categoria;
```

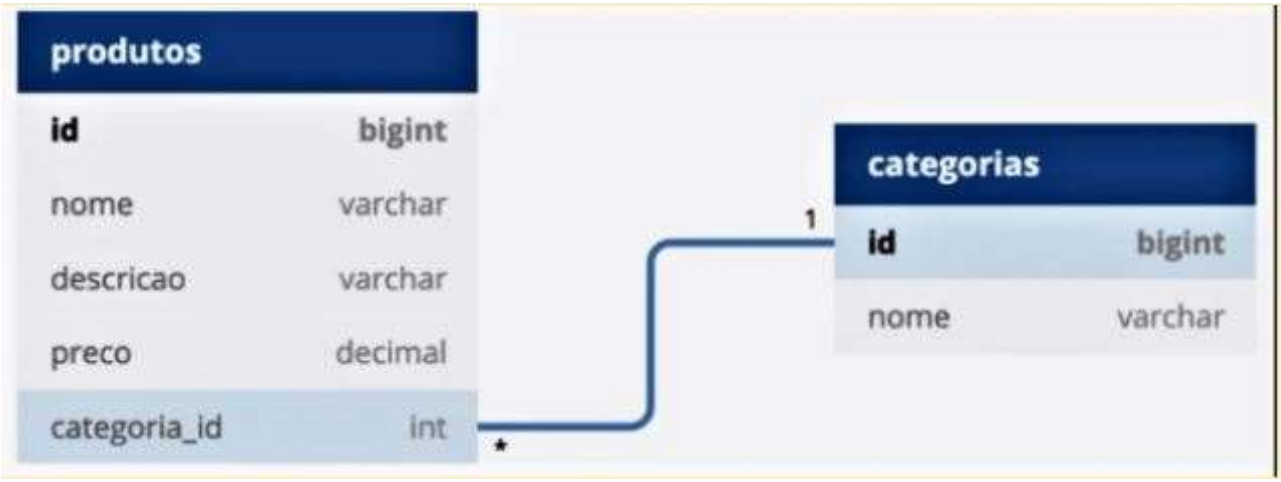
COPIA EL CÓDIGO

- Crear la clase de tipo Enum

```
public enum Categoria {
CELULARES,
TABLETS,
LIBROS;
}
```

COPIA EL CÓDIGO

- Para relacionamientos entre dos entidades



- Se debe añadir la nueva entidad como atributo en la entidad principal con la diferencia que vamos a utilizar la anotación para mapear relaciones de

cardinalidad.

```
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
private Long id;
private String nombre;
private String descripcion;
private BigDecimal precio;
private LocalDate fechaDeRegistro = LocalDate.now();
@ManyToOne
private Categoria categoria;
```

COPIA EL CÓDIGO

- Debe ser creada esa entidad al igual que se hizo con la entidad producto.

```
@Entity
@Table(name="categorias")
public class Categoria {

@Id
@GeneratedValue(strategy=GenerationType.IDENTITY)
private Long id;
private String nombre;

public Categoria() {}

public Categoria(String nombre) {
this.nombre = nombre;
}

public Long getId() {return id;    }
```

```
public void setId(Long id) {this.id = id;}

public String getNombre() {return nombre;}

public void setNombre(String nombre) {this.nombre = nombre;}

}
```

COPIA EL CÓDIGO

- Por último, continuando con la organización del proyecto se crea una clase DAO donde se van a configurar las operaciones de acceso a la base de datos.

```
public class ProdutoDao {

    private EntityManager em;

    public ProdutoDao(EntityManager em) {
        this.em = em;
    }

    public void cadastrar(Produto produto) {
        this.em.persist(produto);
    }

}

public class CategoriaDao{

    private EntityManager em;

public CategoriaDao(EntityManager em) {
```

```
        this.em = em;
        }

        public void cadastrar(Categoria categoria) {
            this.em.persist(categoria);
        }

    }
```

COPIA EL CÓDIGO

Y la clase main ahora se vería de la siguiente forma:

```
public class RegistroDeProducto {

    public static void main(String[] args) {
        Categoria celulares = new Categoria("CELULARES");

        Producto celular = new Producto("Xiaomi Redmi", "Muy bu

        EntityManager em = JPAUtils.getEntityManager();

        ProductoDao productoDao = new ProductoDao(em);
                CategoriaDao categoriaDao = new CategoriaDao(em

        em.getTransaction().begin();

        categoriaDao.guardar(celulares);
        productoDao.guardar(celular);

        em.getTransaction().commit();
        em.close();
    }
```

```
}
```

COPIA EL CÓDIGO